Vorlesungsskript WS 2000/2001

Programmierung

Eine Einführung für Informatiker

Gert Smolka

Fachrichtung Informatik
Universität des Saarlandes, Saarbrücken

Dieses Skript wurde von Prof. Gert Smolka begleitend zur einer Vorlesung an der Universität des Saarlandes im Wintersemester 2000 / 2001 geschrieben. Die Web-Seite dieser Vorlesung finden Sie unter

`http://www.ps.uni-sb.de/courses/prog-ws00/`

Inhaltsverzeichnis

Vorwort

Diese Vorlesung ist eine Einführung in die Programmierung. Sie richtet sich an Informatikstudenten im ersten Semester. Sie behandelt einerseits Datenstrukturen, Algorithmen, Typen, Typabstraktion und Module. Andererseits führt sie in die Theorie und Implementierung von Programmiersprachen ein.

Die Themen der Vorlesung sind vielfältig und gehören zu verschiedenen Teildisziplinen der Informatik. Die Vorlesung ist aber so konzipiert, dass sich ihre Teile schnell zu einem eleganten und in sich schlüssigen Gebäude zusammenfügen, in dem jeder Baustein seinen festen Platz hat.

Als Programmiersprache verwenden wir Standard ML, das auf elegante Weise grundlegende und fortgeschrittene Konzepte der programmiersprachlichen Informatik in ein praktisches Werkzeug umsetzt. Mit Standard ML verschwenden wir keine Zeit für programmiersprachliche Tricks und Ungereimtheiten. Wir legen keinen Wert darauf, alle Details der Programmiersprache Standard ML zu erklären.[1]

Wir beginnen mit mathematischen Datenstrukturen und konzentrieren uns auf die rekursive Definition von Prozeduren (Funktionen) und Typen (Mengen). Standard ML liefert uns eine Sprache, in der wir diese Definitionen präzise aufschreiben können. Statt Papier verwenden wir zum Aufschreiben einen Computer, mit dem wir unsere mathematischen Programme dann auf Konsistenz überprüfen und nach Belieben ausführen können.

Wir lernen, Eigenschaften unserer Programme mit Hilfe von Induktion zu beweisen. Besonders interessieren wir uns dabei für Aussagen über die Korrektheit und die Laufzeit (schnell oder langsam) von Programmen.

Der Weg von mathematischen Programmen zu richtigen Programmen, die sich in die Software-Umgebung eines Computers nahtlos einfügen, ist kurz. Wir benötigen einen Übersetzer sowie Ein- und Ausgabeoperationen für Zeichenreihen.

[1] Wenn Sie mehr über Standard ML wissen wollen, können Sie mit dem Selbststudium eines Lehrbuches beginnen, dessen Hauptziel die Erklärung der Programmiersprache Standard ML ist. Wir empfehlen Ihnen das Lehrbuch von Ullman [20].

Als nächstes betrachten wir Datenstrukturen, deren Zustand sich mit dem Fortschreiten einer Berechnung ändert. Unter anderem versetzen uns solche Datenstrukturen in die Lage, abstrakte Speicher zu programmieren.

Wir beginnen jetzt, größere Programme zu schreiben. Dabei ist es in der Praxis essenziell, Abstraktionen einzuführen und deren konsistente Verwendung mit Hilfe von Typen automatisch zu überprüfen. Gleichzeitig beginnen wir, Programme in klar abgegrenzte Module zu zerlegen, die unabhängig von einander realisiert werden können. Mit Standard ML haben wir eine Programmiersprache, die Typabstraktion und Modularisierung besonders gut unterstützt.

Wir beherrschen jetzt eine Programmiersprache sowie einige Programmiertechniken und können damit nützliche Programme schreiben. Unsere Situation ist vergleichbar mit der eines Kindes, das seine Muttersprache leidlich beherrscht, dem aber die Regeln der Grammatik noch völlig fremd sind. Entsprechend analysieren wir jetzt die Struktur von Teilsprachen von Standard ML, die exemplarisch für alle Programmiersprachen sind. Wir lernen die Unterscheidungen zwischen konkreter und abstrakter Syntax sowie statischer und dynamischer Semantik kennen. Die im ersten Teil der Vorlesung behandelten mathematischen Datenstrukturen und rekursiven Definitionstechniken versetzen uns in die Lage, Syntax und Semantik von Programmiersprachen präzise zu definieren. Der Kreis beginnt sich zu schließen.

Wir haben jetzt die Voraussetzungen, um grundlegende Techniken für die Realisierung von Programmiersprachen auf Computern zu lernen. Diese erfolgt mit Programmen. Wir lernen, wie man einen einfachen Interpreter schreibt. Danach betrachten wir eine virtuelle Maschine mit automatischer Speicherverwaltung, die eine hardwarenahe Programmiersprache realisiert. Die Programme für die virtuelle Maschine erzeugen wir mit einem Übersetzer für eine Teilsprache von Standard ML. Den Interpreter, die virtuelle Maschine und den Übersetzer schreiben wir in Standard ML. Der Kreis ist geschlossen.

Danksagung Dieses Skript entstand begleitend zu einer Vorlesung, die ich an der Universität des Saarlandes bisher dreimal gehalten habe (WS 98/99, WS 99/00, WS 00/01).

Zuerst möchte ich Andreas Rossberg und Thorsten Brunklaus danken, die die Vorlesung als Assistenten betreut haben. Sie haben durch Diskussionen und Korrekturen beständig zu diesem Skript beigetragen. Weiter geht mein Dank an Marco Kuhlmann, der mir bei der typografischen Erstellung des Skripts geholfen hat und sich dabei schnell zum LaTeX-Experten entwickelt hat (LaTeX ist die Dokumentenbeschreibungssprache, in der das Skript geschrieben ist). Schließlich danke ich

den Übungsgruppenleitern und Hörern der Vorlesung für ihre Fragen, Anregungen und das Aufdecken von Fehlern.